Master M2 MSIAM

Learning Probabilities and Causality

Graded Lab on Variational Autoencoders

The lab report is due on December 19th, 2023 and consists of two items:

- The Jupyter Notebook file with your code (to answer section 2.3).
- A PDF with the plots and comments answering sections 2.4 and 2.5.
  This PDF should NOT exceed 3 pages.

Send a ZIP file by e-mail to `xavier.alameda-pineda@inria.fr`.

2 Variational Autoencoders for Speech Modeling

2.1 Introduction

In this lab work we are going to model the speech signal using variational autoencoders, and derived models. We have prepare the data for you, and you will have access to sequences of magnitude spectrograms. In practice this means that your data will be sequences of $T = 150$ vectors of dimension $F = 257$. These vectors are obtained applying the short-time Fourier transform (which leads to a complex-valued vector) and then computing the entry-wise modulus, therefore we write: $\boldsymbol{x}_n \in \mathbb{R}^F$, where $n = 1, \ldots, N$ is the sample index, and N is the total number of samples in the training or evaluation sets. Although we have used a large-scale training set to provide a pre-trained model for some of the questions, you will be using two small datasets from two different speakers, Speaker A and Speaker B.

The generative model consists of a latent Z-dimensional variable (also called code), denoted by $\boldsymbol{z} \in \mathbb{R}^Z$. In our experiments we will use $Z = 16$. The probabilistic model writes:

$$\boldsymbol{z} \sim \mathcal{N}(\boldsymbol{z}; \boldsymbol{0}, \boldsymbol{I}) \qquad \text{and} \qquad \boldsymbol{x}|\boldsymbol{z} \sim \mathcal{N}_c(\boldsymbol{x}; \boldsymbol{0}, \boldsymbol{\nu}_{\boldsymbol{\theta}}(\boldsymbol{z})), \tag{1}$$

where $\boldsymbol{0}$ and $\boldsymbol{I}$ are the zero-valued vector and identity matrix of appropriate dimensions respectively and $\mathcal{N}_c$ denotes the circular complex Gaussian defined as:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) = \mathcal{N}_c(\boldsymbol{x}; \boldsymbol{0}, \boldsymbol{\nu}_{\boldsymbol{\theta}}(\boldsymbol{z})) = \frac{1}{\pi^F |\boldsymbol{\nu}_{\boldsymbol{\theta}}(\boldsymbol{z})|} \exp\left(-\boldsymbol{x}^\top \boldsymbol{\nu}_{\boldsymbol{\theta}}(\boldsymbol{z})^{-1} \boldsymbol{x}\right). \tag{2}$$

where $\boldsymbol{\theta}$ denote the parameters of the decoder network.

Similarly to the standard VAE formulation seen in class, the covariance matrix depends on the latent code $\boldsymbol{z}$. Differently the standard formulation, the mean is assumed to be zero. In addition, as done in class, we will be implementing a diagonal covariance matrix with log-variances. Formally:

$$\boldsymbol{\nu}_{\boldsymbol{\theta}}(\boldsymbol{z}) = \begin{pmatrix} \exp(\eta_{1,\boldsymbol{\theta}}(\boldsymbol{z})) & 0 & \dots \\ 0 & \exp(\eta_{2,\boldsymbol{\theta}}(\boldsymbol{z})) & \dots \\ \vdots & \vdots & \ddots \end{pmatrix} \tag{3}$$

In other works, the decoder which is the part of the network implementing the generative model, will input $\boldsymbol{z}$ and output $\boldsymbol{\eta}_{\boldsymbol{\theta}}(\boldsymbol{z}) = [\eta_{1,\boldsymbol{\theta}}(\boldsymbol{z}), \dots, \eta_{F,\boldsymbol{\theta}}(\boldsymbol{z})] \in \mathbb{R}^F$. Recall that $\boldsymbol{\theta}$ are the parameters of the decoder network.

The inference model approximates the posterior of the latent code with a Gaussian distribution:

$$p(\boldsymbol{z}|\boldsymbol{x}) \approx q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}; \boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}), \boldsymbol{\nu}_{\boldsymbol{\phi}}(\boldsymbol{x})), \tag{4}$$

where $\boldsymbol{\phi}$ are the parameters of the so-called encoder network. The covariance matrix of the approximate posterior distribution follows the same structure and rationale as the one of the generative model (i.e. we estimate a vector of log-variances that form a diagonal matrix). As a consequence, the encoder inputs $\boldsymbol{x}$ and outputs $[\boldsymbol{\mu}_{\boldsymbol{\phi}}(\boldsymbol{x}), \boldsymbol{\eta}_{\boldsymbol{\phi}}(\boldsymbol{x})] = [\mu_{1,\boldsymbol{\phi}}(\boldsymbol{x}), \dots, \mu_{Z,\boldsymbol{\phi}}(\boldsymbol{x}), \eta_{1,\boldsymbol{\phi}}(\boldsymbol{x}), \dots, \eta_{Z,\boldsymbol{\phi}}(\boldsymbol{x})] \in \mathbb{R}^{2Z}$.

The loss consists of two terms: reconstruction and regularisation. More formally:

$$\mathcal{L}(\boldsymbol{x}_{1:N}, \boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{n=1}^{N} \mathbb{E}_{q_{\boldsymbol{\phi}}(\boldsymbol{z}_n|\boldsymbol{x}_n)}\{\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n|\boldsymbol{z}_n)\} - \mathcal{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}_n|\boldsymbol{x}_n)\|p(\boldsymbol{z}_n)). \tag{5}$$

This is a lower-bound of the log-likelihood that needs to be **maximised**. If you use it within a minimisation framework, you need to change the sign of the expression. Moreover, since the first expectation cannot be taken analytically, we need to approximate with a sample:

$$\mathcal{L}(\boldsymbol{x}_{1:N}, \boldsymbol{\theta}, \boldsymbol{\phi}) = \sum_{n=1}^{N} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n|\hat{\boldsymbol{z}}_n) - \mathcal{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}_n|\boldsymbol{x}_n)\|p(\boldsymbol{z}_n)), \qquad \hat{\boldsymbol{z}}_n \sim q_{\boldsymbol{\phi}}(\boldsymbol{z}_n|\boldsymbol{x}_n). \tag{6}$$

2.2 Pre-implementation work

1. Replace the expression of $\boldsymbol{\nu}_{\boldsymbol{\theta}}(\boldsymbol{z})$ into the generative model $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$ to compute the sampled log-likelihood $\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_n|\hat{\boldsymbol{z}}_n)$ for the reconstruction term of the loss.

2. Find the formula for the Kullback-Leibler divergence between two multivariate Gaussians, and use it to compute the regularisation term of the loss

$$\mathcal{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}_n|\boldsymbol{x}_n)\|p(\boldsymbol{z}_n)), \tag{7}$$

taking into account the structure of $\boldsymbol{\nu}_{\boldsymbol{\phi}}(\boldsymbol{x})$.

3. Write the re-parametrisation trick as a function of the parameters provided by the encoder, that is the mean and the log-variance.

2.3 Implementing VAEs

We will be implementing a VAE with the following specifications. The encoder has four dense layers of output dimensions (256, 128, 64, 32) activated with a hyperbolic tangent. After these four layers there is a linear layer to output the mean and the log-variance of $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$. The decoder has also four dense layers with output dimensions (32, 64, 128, 256) activated with a hyperbolic tangent. After these four layers, there is a linear layer to output the log-variance of $p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z})$.

Your implementation work has several steps:

1. Implement the structure of the encoder and decoder in the `build` function. In the code provided, we split the non-linear layers (that should be implemented in `mlp_x_z` and `mlp_z_x`) from the rest.

2. Implement the `generation_x` function (i.e. computing the log-variance of $p(\boldsymbol{s}|\boldsymbol{z})$.

3. Implement the `reparametrization` trick as was discussed in class, and you rewrote in the pre-implementation work, in the `reparametrization` function.

4. Implement the `inference` function, to compute the mean and log-variance of $p(\boldsymbol{z}|\boldsymbol{s})$ and sample from it using the `reparametrization` function.

5. Implement the loss with the formulae that you have derived in the pre-implementation work, in the `get_loss` function.

Check your implementation with the short test code provided.

2.4 Training and evaluation

In this section we will use the data from Speaker A and Speaker B jointly.

1. Train the implemented architecture for 30 epochs using the provided code. Plot the joint loss as well as the reconstruction and regularisation terms separately.

2. Repeat the training from scratch five times. Plot the average and standard deviation of the loss over the epochs.

2.5 Fine-tuning a pre-trained model

In this section we will use the data from Speaker A and Speaker B separately. Download the pre-trained model. For your information, this model has been trained with data collected from around 100 different speakers (which is roughly 9 hours of speech) and for 164 epochs. The pre-trained model is quite different from what you have trained in the previous section.

1. Read the code provided for pre-processing and comment it.

2. Load the model and compute the loss (without any fine-tuning) on the samples of Speaker A and of Speaker B.

3. Fine-tune the pre-trained model on the data of Speaker A and of Speaker B separately, for 20 epochs. You must obtain two models (Model A and Model B). Evaluate each model with the data of the corresponding speaker. What do you observe if you compare it to the previous exercise?

4. Evaluate now Model A with the data of Speaker B, and vice-versa. What do you observe when you compare it with the previous two situations?